

Pembelajaran Dalam (Deep Learning)

Chapter 3
TensorFlow-1: Basic Utility

Genap TA 2022/2023

FYS, ITQ | Maret 2023

1. Pendahuluan Pembelajaran Mesin dan Pembelajaran Dalam (CLO 1)
2. Programming Python: Dasar Numpy, Pandas, Matplotlib, dll (CLO 3)
3. Dasar Regresi dan Klasifikasi (CLO 1)
4. Artificial Neural Networks (ANN) (CLO 2/3)
5. Multi-Layer Perceptron (MLP) (CLO 2/3)
- 6. TensorFlow (CLO 3)**
7. TensorFlow (+Kinerja Machine Learning) (CLO 3/1)
8. TensorFlow Lanjutan (CLO 3)
9. Convolutional Neural Networks (CNN) (CLO 2/3)
10. CNN Lanjutan (CLO 2/3)
11. Recurrent Neural Networks (RNN) (CLO 2/3)
12. RNN Lanjutan (CLO 2/3)
13. Generative Adversarial Networks (GANs) (CLO 2/3)
14. GANs lanjutan (CLO 2/3)
15. Aplikasi Object Detection (CLO 2/3)
16. UAS (CLO 2/3)

TensorFlow-1: Basic

Course Learning Outcomes (CLO)

- **CLO 3:** Mahasiswa dapat mengimplementasikan algoritma pembelajaran dalam yang populer menggunakan perangkat lunak

TensorFlow-1: Basic

Tujuan Perkuliahan

- Mempelajari library popular untuk *Deep Learning* yaitu **TensorFlow**
- Mengimplementasikan TensorFlow pada multilayer neural networks

TensorFlow-1: Basic

Outline:

A. TensorFlow dan Kinerja Training

- Tantangan Kinerja
- Apa itu TensorFlow

B. Langkah-Langkah Awal untuk TensorFlow

- Membuat Tensor pada TensorFlow
- Memanipulasi Tipe Data dan Ukuran Tensor
- Operasi Matematika pada Tensor
- Split, stack dan Concatenate untuk Tensor-Tensor

C. Membangun Input Pipelines dengan tf.data

- Membuat Dataset TensorFlow
- Kombinasi Dua Tensor menjadi Joint Dataset
- Shuffle, Batch dan Repeat
- Membuat Dataset dari File-File di Disk Lokal
- Fetching Dataset dari Library tensorflow_datasets

TensorFlow-1: Basic

- Materi TensorFlow terbagi menjadi 3 bagian
 1. TensorFlow-1: Basic (**dijelaskan di pertemuan ini**)
 2. TensorFlow-2: Model Predisi Neural Networks
 3. TensorFlow-3: Eksplorasi Lebih Dalam

TensorFlow-1: Basic

A. TensorFlow dan Kinerja Training

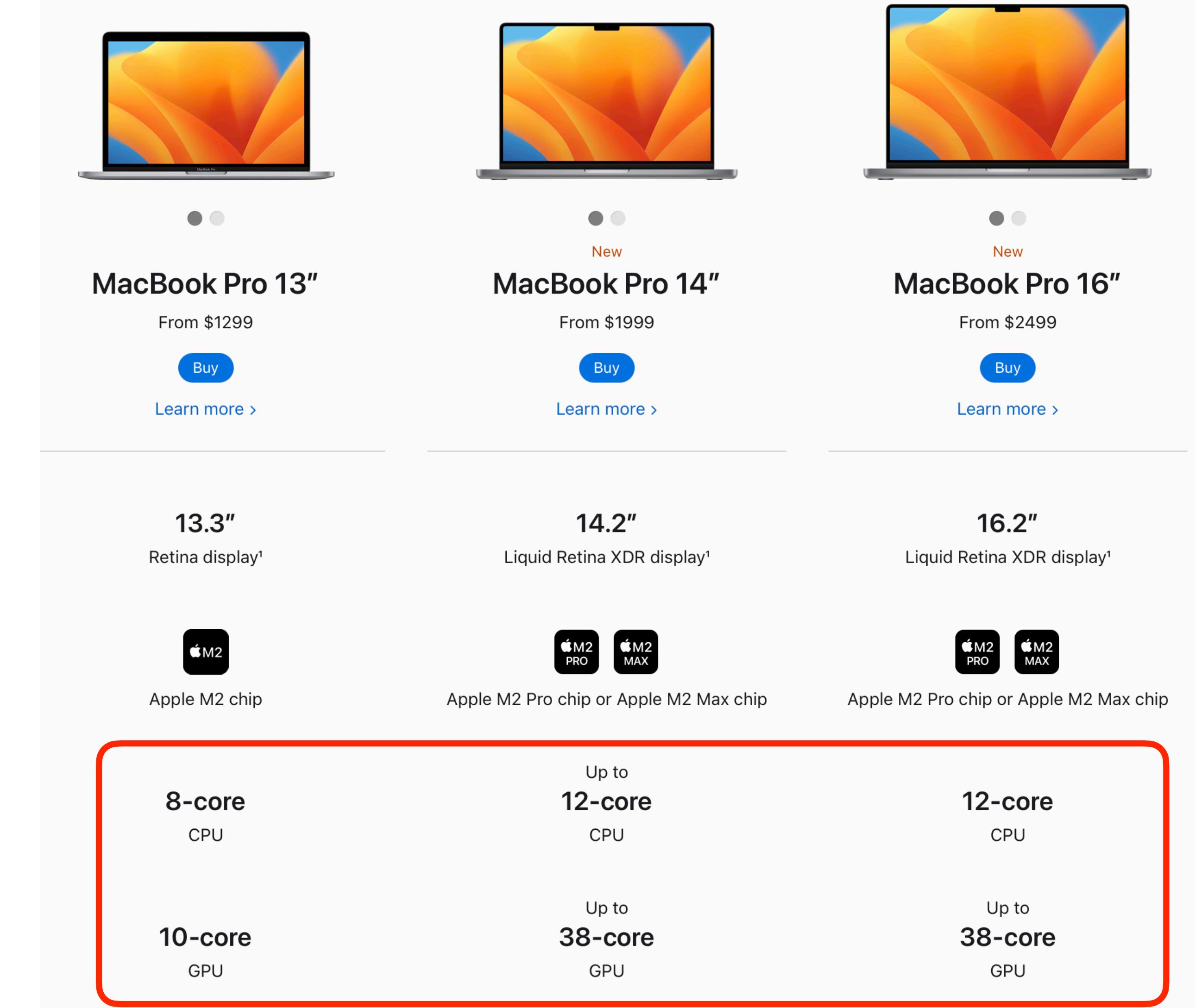
TensorFlow dapat memperbaiki kinerja Machine Learning (mempercepat) secara signifikan

TensorFlow-1: Basic

A.1. Tantangan Kinerja

- Kinerja prosessor komputer meningkat tiap tahun, sehingga dapat digunakan untuk training pada *learning systems* yang kompleks
- Desktop dengan **multi-cores** semakin murah, tetapi desktop yang paling canggih jarang yang dilengkapi dengan jumlah core lebih dari 8 atau 16.

Contoh: Spesifikasi Jumlah Core pada Apple



TensorFlow-1: Basic

A.1. Tantangan Kinerja

- Dataset MNIST dengan ukuran pixel $28 \times 28 = 784$ features dan 1 hidden layer dengan 100 unit
 - ➡ Harus melakukan optimasi parameter pembobot sekitar 80 ribu ($[784 \times 100 + 100] + [100 \times 10] + 10 = 79.510$)
 - ➡ Jika ukuran piksel image-image yang digunakan lebih tinggi dan jumlah hidden layer lebih banyak, maka akan terjadi ledakan jumlah parameter
- Kondisi di atas tidak mungkin dilakukan oleh unit pemrosesan tunggal, tetapi menggunakan **Graphical Processing Units** (GPU)

TensorFlow-1: Basic

A.1. Tantangan Kinerja

- GPU adalah kluster komputer kecil dalam sebuah mesin dengan kemampuan komputasi tinggi
- GPU lebih murah, kira-kira 65 % harga CPU, tetapi kurang lebih mempunyai 272 kali lebih banyak jumlah core
- GPU mampu melakukan 10 kali lebih cepat perhitungan floating point disbanding CPU

Perbandingan CPU dan GPU		
Specifications	Intel® Core™ i9-9960X X-series Processor	NVIDIA GeForce® RTX™ 2080 Ti
Base Clock Frequency	3.1 GHz	1.35 GHz
Cores	16 (32 threads)	4352
Memory Bandwidth	79.47 GB/s	616 GB/s
Floating-Point Calculations	1290 GFLOPS	13400 GFLOPS
Cost	~ \$1700.00	~ \$1100.00

TensorFlow-1: Basic

A.1. Tantangan Kinerja

- Menuliskan kode untuk target GPU tidak sesederhana mengeksekusi kode Python dalam interpreter.
 - ➡ Banyak package special dapat digunakan pada GPU, misalkan:
 - **CUDA** (<https://en.wikipedia.org/wiki/CUDA>)
 - **OpenCL** (<https://en.wikipedia.org/wiki/OpenCL>)
- Menulis kode pada CUDA dan OpenCL tidak mudah dan mungkin tidak menyediakan *environment* untuk *programming* yang nyaman
 - ➡ TensorFlow memungkinkan kita untuk menuliskan kode-kode pada CUDA atau OpenCL dalam mengimplementasikan dan mengeksekusi algoritma-algoritma Machine Learning

TensorFlow-1: Basic

A.1. Tantangan Kinerja

NVIDIA CUDA on Jetson Nano



Jetson Nano Developer Kit

GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (not included)
Video Encode	4K @ 30 4x 1080p @ 30 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 2x 4K @ 30 8x 1080p @ 30 18x 720p @ 30 (H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI and display port
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I ² C, I ² S, SPI, UART
Mechanical	69 mm x 45 mm, 260-pin edge connector

TensorFlow-1: Basic

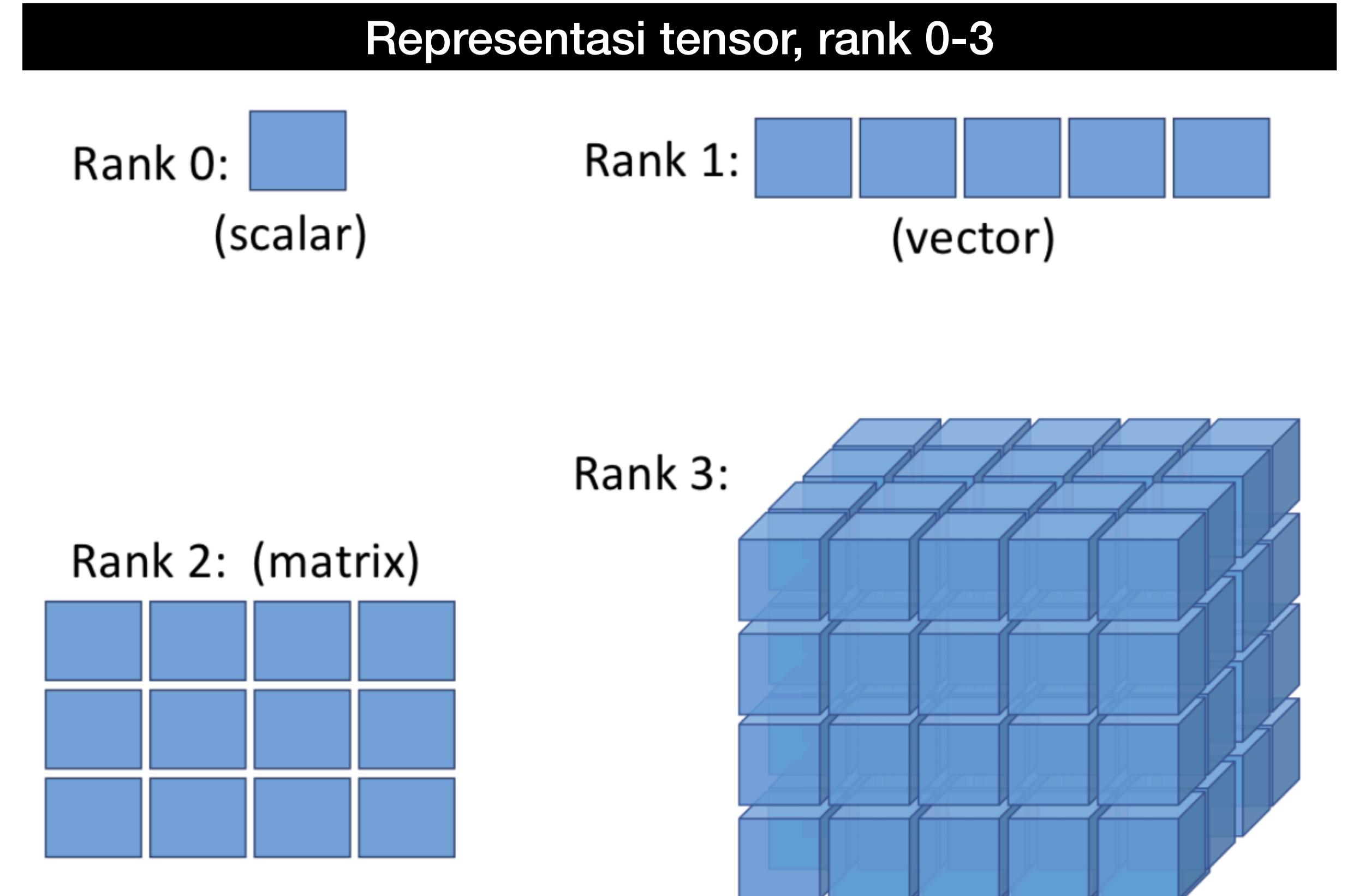
A.2. Apa itu TensorFlow

- TensorFlow adalah antarmuka (interface) pemrograman yang scalable dan multiplatform untuk mengimplementasikan dan mengeksekusi algoritma-algoritma machine learning
- TensorFlow dibangun oleh peneliti dan Insinyur dari Tim Google Brain dan juga kontribusi dari komunitas open source
- Banyak peneliti dan praktisi dari akademia dan industri mengadaptasi TensorFlow untuk membangun solusi-solusi deep learning
- TensorFlow dapat dieksekusi di CPUs maupun GPUs, tapi terbaik jika digunakan GPUs
- TensorFlow mendukung GPU berbasis CUDA dan OpenCL
- **Application Programming Interface** (API) Python dari TensorFlow saat ini merupakan API paling komplit

TensorFlow-1: Basic

A.2. Apa itu TensorFlow

- TensorFlow berbasis **tensor**, yang dapat dipandang sebagai generalisasi dari skalar, vektor, matriks dan selanjutnya.
- Secara konkret: **Skalar = rank-0 tensor, Vektor = rank-1 tensor, matriks = rank-2 tensor, dan matriks yang ditumpuk pada dimensi ketiga sebagai rank-3 tensor**
- TensorFlow menyimpan harga-harga pada array-array `NumPy`, dan tensor menyediakan referensi pada array tersebut.



TensorFlow-1: Basic

B. Langkah-langkah awal untuk TensorFlow

- Mempelajari langkah awal dalam penggunaan API *low level* dari TensorFlow
- Mempelajari bagaimana membuat dan memanipulasi tensor pada TensorFlow, seperti merubah ukuran, tipe data dll
- Untuk instalasi TensorFlow tidak akan dijelaskan, dapat diakses melalui link berikut: <https://www.tensorflow.org/install>

TensorFlow-1: Basic

B.1. Membuat Tensor pada TensorFlow

- Melihat versi TensorFlow

```
import tensorflow as tf
print('TensorFlow version:', tf.__version__)
import numpy as np

np.set_printoptions(precision=3)
```

TensorFlow version: 2.9.1

- Membuat sebuah tensor dari list atau array NumPy menggunakan fungsi `tf.convert_to`

```
a = np.array([1, 2, 3], dtype=np.int32)
b = [4, 5, 6]

t_a = tf.convert_to_tensor(a)
t_b = tf.convert_to_tensor(b)

print(t_a)
print(t_b)
```

```
tf.Tensor([1 2 3], shape=(3,), dtype=int32)
tf.Tensor([4 5 6], shape=(3,), dtype=int32)
```

TensorFlow-1: Basic

B.1. Membuat Tensor pada TensorFlow

- Melihat properties dari tensor, mis. shape

```
t_ones=tf.ones((2,3))  
t_ones.shape
```

```
TensorShape([2, 3])
```

- Mendapatkan akses harga kemana tensor menunjuk, gunakan metode .numpy()

```
t_a.numpy()
```

```
array([1, 2, 3], dtype=int32)
```

```
const_tensor=tf.constant([1.2,5,np.pi],dtype=tf.float32)  
print(const_tensor)
```

```
tf.Tensor([1.2 5. 3.142], shape=(3,), dtype=float32)
```

```
const_tensor.numpy()
```

```
array([1.2 , 5. , 3.142], dtype=float32)
```

- Membuat tensor harga konstan

TensorFlow-1: Basic

B.2. Memanipulasi Tipe Data dan Ukuran Tensor

- Fungsi `tf.cast()` dapat digunakan untuk merubah tipe data dari tensor

```
t_a_new = tf.cast(t_a,tf.int64)
print(t_a_new.dtype)
```

```
<dtype: 'int64'>
```

- Transpose sebuah tensor

```
t = tf.random.uniform(shape=(3, 5))

t_tr = tf.transpose(t)
print(t.shape, ' --> ', t_tr.shape)
```

```
(3, 5) --> (5, 3)
```

TensorFlow-1: Basic

B.3. Operasi Matematika pada Tensor

- Menerapkan operasi matematika pada tensor diperlukan untuk membangun model-model machine learning (terutama aljabar linier)
- Menginisiasi dua tensor random, satu terdistribusi uniform pada range $[-1,1]$, dan dengan distribusi normal

```
tf.random.set_seed(1)
t1 = tf.random.uniform(shape=(5, 2), minval=-1.0, maxval=1.0)
t2 = tf.random.normal(shape=(5, 2), mean=0.0, stddev=1.0)
```

Perhatikan bahwa t1 dan t2 mempunyai ukuran (shape) yang sama

TensorFlow-1: Basic

B.3. Operasi Matematika pada Tensor

- Perkalian antar elemen kedua matriks

```
t3 = tf.multiply(t1, t2).numpy()  
print(t3)
```

```
[[ -0.27 -0.874]  
 [ -0.017 -0.175]  
 [ -0.296 -0.139]  
 [ -0.727  0.135]  
 [ -0.401  0.004]]
```

- Menghitung mean sepanjang samba tertentu, gunakan `tf.math.reduce_mean()`

```
t4 = tf.math.reduce_mean(t1, axis=0) # untuk mean sepanjang kolom gunakan axis=1  
print(t4)
```

```
tf.Tensor([0.09  0.207], shape=(2,), dtype=float32)
```

TensorFlow-1: Basic

B.3. Operasi Matematika pada Tensor

- Perkalian matriks antara t_1 dan t_2 yaitu $t_1 \times t_2^T$ dimana $(\cdot)^T$ adalah transpose

```
t5 = tf.linalg.matmul(t1, t2, transpose_b=True)
print(t5.numpy())
```

```
[[ -1.144  1.115 -0.87  -0.321  0.856]
 [ 0.248 -0.191  0.25   -0.064 -0.331]
 [-0.478  0.407 -0.436  0.022  0.527]
 [ 0.525 -0.234  0.741 -0.593 -1.194]
 [-0.099  0.26   0.125 -0.462 -0.396]]
```

- Sedangkan untuk $t_1^T \times t_2$

```
t6 = tf.linalg.matmul(t1, t2, transpose_a=True)
print(t6.numpy())
```

```
[[ -1.711  0.302]
 [ 0.371 -1.049]]
```

TensorFlow-1: Basic

B.3. Operasi Matematika pada Tensor

- Fungsi `tf.norm()` untuk menghitung norm L^p . Misal untuk norm L^2 dari `t1`

```
norm_t1 = tf.norm(t1, ord=2, axis=1).numpy()  
print(norm_t1)
```

[1.046 0.293 0.504 0.96 0.383]

- Untuk memastikan betul, bandingkan dengan hasil NumPy berikut

```
np.sqrt(np.sum(np.square(t1), axis=1))
```

array([1.046, 0.293, 0.504, 0.96 , 0.383], dtype=float32)

TensorFlow-1: Basic

B.4. Split, Stack dan Concatenate Tensor-Tensor

- Pada bagian ini akan melihat operasi TensorFlow untuk
 - memisahkan (split) sebuah tensor
 - Menumpuk (stack) tensor-tensor
 - Menyambung urutkan tensor-tensor (concatenate)
- Diasumsikan kita mempunyai satu tensor dan ingin dibagi menjadi 2 atau lebih tensor-tensor

TensorFlow-1: Basic

B.4. Split, Stack dan Concatenate Tensor-Tensor

- Sebuah tensor dengan ukuran (size) 6, dibagi ke dalam sebuah list berisi 3 tensor (num_or_size_splits = 3)

```
tf.random.set_seed(1)
t = tf.random.uniform((6,))
print(t.numpy())
```

```
[0.165 0.901 0.631 0.435 0.292 0.643]
```

```
t_splits = tf.split(t, num_or_size_splits =3)
[item.numpy() for item in t_splits]
```

```
[array([0.165, 0.901], dtype=float32),
 array([0.631, 0.435], dtype=float32),
 array([0.292, 0.643], dtype=float32)]
```

TensorFlow-1: Basic

B.4. Split, Stack dan Concatenate Tensor-Tensor

- Jika ukuran 6×2 dan dibagi menjadi 2 tensor sepanjang baris (axis=0)

```
t = tf.random.uniform((6,2))
print(t.numpy())
t_splits = tf.split(t, num_or_size_splits =2, axis=0)
[item.numpy() for item in t_splits]
```

```
[[0.51  0.444]
 [0.409 0.992]
 [0.689 0.346]
 [0.436 0.601]
 [0.457 0.753]
 [0.188 0.549]]
```

```
[array([[0.51 , 0.444],
       [0.409, 0.992],
       [0.689, 0.346]], dtype=float32),
 array([[0.436, 0.601],
       [0.457, 0.753],
       [0.188, 0.549]], dtype=float32)]
```

TensorFlow-1: Basic

B.4. Split, Stack dan Concatenate Tensor-Tensor

- Kita dapat menyediakan ukuran-ukuran split yang berbeda (daripada mendefinisikan jumlah split). Tensor dengan ukuran 5 menjadi tensor dengan ukuran 3 dan 2

```
tf.random.set_seed(1)
t = tf.random.uniform((5,4))
print(t.numpy())
t_splits = tf.split(t, num_or_size_splits=[2,2], axis=1) #untuk split kolom
# t_splits= tf.split(t, num_or_size_splits=[3,2], axis=0) #untuk split baris
[item.numpy() for item in t_splits]
```

```
[[0.165 0.901 0.631 0.435]
 [0.292 0.643 0.976 0.435]
 [0.66  0.605 0.637 0.614]
 [0.889 0.628 0.532 0.026]
 [0.441 0.253 0.886 0.887]]
```

```
[array([[0.165,  0.901],
       [0.292,  0.643],
       [0.66 ,  0.605],
       [0.889,  0.628],
       [0.441,  0.253]], dtype=float32),
 array([[0.631,  0.435],
       [0.976,  0.435],
       [0.637,  0.614],
       [0.532,  0.026],
       [0.886,  0.887]], dtype=float32)]
```

TensorFlow-1: Basic

B.4. Split, Stack dan Concatenate Tensor-Tensor

- Jika ada tensor 1D **A** (ukuran 3 dan berisi semua 1) dan tensor lain 1D **B** (ukuran 2 berisi semua 0), maka disambung-urutkan (concatenate) menjadi 1D **C** (ukuran 5)

```
A = tf.ones((3,))  
B = tf.zeros((3,))  
C = tf.concat([A, B], axis=0)  
print(C.numpy())
```

[1. 1. 1. 0. 0. 0.]

- Jika **A** dan **B** keduanya berukuran 5 maka dapat ditumpuk (stack), menjadi 2D

```
A = tf.ones((3,))  
B = tf.zeros((3,))  
S = tf.stack([A, B], axis=1)  
print(S.numpy())
```

[[1. 0.]
 [1. 0.]
 [1. 0.]]

TensorFlow-1: Basic

B.4. Split, Stack dan Concatenate Tensor-Tensor

- API dari TensorFlow mempunyai banyak operasi yang dapat digunakan untuk membuat sebuah model, memproses data dll.
- Untuk full operasi dan fungsi yang bisa digunakan, maka bisa dilihat di documentais TensorFlow pada link berikut:

<https://www.tensorflow.org/learn>

TensorFlow-1: Basic

C. Membangun input pipelines menggunakan tf.data

- Ketika dataset besar untuk sebuah memori komputer, maka kanang perlu *load* data dari devais penyimpanan utama dalam bentuk *chunks*, atau *per-batch*
- Selain itu, boleh jadi kita mengkonstruksi pipeline pemrosesan data, seperti mean centering dan scaling
- Menerapkan fungsi-fungsi preprocessing secara manual setiap saat akan membuat tidak praktis
- TensorFlow menyediakan *class* special untuk melakukan konstuksi pipeline dari preprocessing secara efisien dan mudah
- Pada bagian ini, akan dibahas overview dari metode-metode berbeda untuk konstruktiv TensorFlow Dataset, termasuk transforms dataset dan langkah-langkah umum pada preprocessing

TensorFlow-1: Basic

C.1. Membuat Dataset TensorFlow dari Tensor-Tensor yang ada

- Jika data telah ada dalam bentuk tensor, list Python, atau array-array NumPy, kita dapat membuat dataset secara mudah menggunakan fungsi `tf.data.Dataset.from_tensor_slices()`
- Fungsi tersebut mengembalikan sebuah objek dengan class `Dataset`, dengan elemen-elemen individunya dapat diiterasi pada input dataset
- Kode berikut membuat dataset dari sebuah list

```
a = [1.2, 3.4, 7.5, 4.1, 5.0, 1.0]
ds = tf.data.Dataset.from_tensor_slices(a)
print(ds)
```

```
<TensorSliceDataset element_spec=TensorSpec(shape=(), dtype=tf.float32,
name=None)>
```

TensorFlow-1: Basic

C.1. Membuat Dataset TensorFlow dari Tensor-Tensor yang ada

- Untuk iterasi melalui dataset pada setiap entri-nya

```
for item in ds:  
    print(item)
```

```
tf.Tensor(1.2, shape=(), dtype=float32)  
tf.Tensor(3.4, shape=(), dtype=float32)  
tf.Tensor(7.5, shape=(), dtype=float32)  
tf.Tensor(4.1, shape=(), dtype=float32)  
tf.Tensor(5.0, shape=(), dtype=float32)  
tf.Tensor(1.0, shape=(), dtype=float32)
```

- Membuat batch-batch dari dataset, dengan ukuran 3

```
ds_batch = ds.batch(3)  
for i, elem in enumerate(ds_batch, 1):  
    print('batch {}:{}'.format(i), elem.numpy())
```

```
batch 1: [1.2 3.4 7.5]  
batch 2: [4.1 5. 1. ]
```

TensorFlow-1: Basic

C.1. Membuat Dataset TensorFlow dari Tensor-Tensor yang ada

- Kode di atas membuat dua batch dari dataset, dimana tiga elemen masuk ke batch #1 dan elemen-elemen sisa masuk ke batch #2
- Metode `.batch()` mempunyai opsi argumen `drop_remainder` (default-nya `false`) yang berguna ketika jumlah elemen-elemen pada tensor tidak dibagi dengan ukuran batch yang diinginkan

TensorFlow-1: Basic

C.2. Mengkombinasikan Dua Tensor menjadi joint dataset

- Misalkan kita ingin mengkombinasikan dua tensor, yaitu tensor feature dan tensor label ke dalam satu dataset, sehingga kita dapat mengambil elemen-elemen dari tensor dalam bentuk tuple
- Misal ada dua tensor t_x (menyimpan harga feature ukuran 3) dan t_y (berisi label-label class)

```
tf.random.set_seed(1)
t_x = tf.random.uniform([4, 3], dtype=tf.float32)
t_y = tf.range(4)
```

TensorFlow-1: Basic

C.2. Mengkombinasikan Dua Tensor menjadi joint dataset

- Kita ingin membuat joint dataset dari kedua tensor tersebut

```
ds_x = tf.data.Dataset.from_tensor_slices(t_x)
ds_y = tf.data.Dataset.from_tensor_slices(t_y)
ds_joint = tf.data.Dataset.zip((ds_x, ds_y))
for example in ds_joint:
    print(' x: ', example[0].numpy(),
          ' y: ', example[1].numpy())
```

```
x: [0.165 0.901 0.631]    y:  0
x: [0.435 0.292 0.643]    y:  1
x: [0.976 0.435 0.66 ]    y:  2
x: [0.605 0.637 0.614]    y:  3
```

TensorFlow-1: Basic

C.2. Mengkombinasikan Dua Tensor menjadi joint dataset

- Alternatif lain adalah

```
## metode alternatif:  
ds_joint = tf.data.Dataset.from_tensor_slices((t_x, t_y))  
for example in ds_joint:  
    print(' x: ', example[0].numpy(), ' y: ', example[1].numpy())
```

```
x: [0.165 0.901 0.631]      y:  0  
x: [0.435 0.292 0.643]      y:  1  
x: [0.976 0.435 0.66 ]     y:  2  
x: [0.605 0.637 0.614]     y:  3
```

TensorFlow-1: Basic

C.2. Mengkombinasikan Dua Tensor menjadi joint dataset

- Implementasi transformasi pada setiap elemen dari sebuah dataset, misalkan feature-scaling untuk penskalaan harga-harga elemen pada t_x pada range $[-1,1]$ menjadi $[0,1]$ berdasarkan distribusi uniform

```
ds_trans = ds_joint.map(lambda x, y: (x*2-1.0, y))

for example in ds_trans:
    print(' x: ', example[0].numpy(), ' y: ', example[1].numpy())
```

```
x: [-0.67  0.803  0.262]  y:  0
x: [-0.131 -0.416  0.285]  y:  1
x: [ 0.952 -0.13   0.32 ]  y:  2
x: [0.21  0.273  0.229]  y:  3
```

TensorFlow-1: Basic

C.3. Shuffle, Batch dan Repeat

- Kadang untuk melakukan training (mis. SGD) penting untuk menggunakan batch-batch yang dikocok secara arak (*randomly shuffled batch*). Akan diperlihatkan cara shuffle dan re-iterasi pada dataset
- Membuat versi yang telah di-shuffle dari dataset `ds_joint`

```
tf.random.set_seed(1)
ds = ds_joint.shuffle(buffer_size=len(t_x))
for example in ds:
    print(' x: ', example[0].numpy(), ' y: ', example[1].numpy())
```

```
x: [0.976 0.435 0.66 ]  y: 2
x: [0.435 0.292 0.643]  y: 1
x: [0.165 0.901 0.631]  y: 0
x: [0.605 0.637 0.614]  y: 3
```

Baris telah di-shuffle tanpa mempengaruhi korespondensi satu-ke-satu antara entri-entri x dan y

TensorFlow-1: Basic

C.3. Shuffle, Batch dan Repeat

- Metode `.shuffle()` perlu argumen `buffer_size` yang menyatakan banyak elemen-elemen pada dataset yang digrupkan secara bersamaan sebelum di-shuffle
- Untuk menjamin randomisasi shuffle pada tiap epoch, kita bisa memilih ukuran buffer sama dengan jumlah training examples, contoh kode di atas menggunakan `buffer_size=len(t_x)`
- Membagi dataset menjadi beberapa batch untuk training dengan metode `.batch()`

```
ds = ds_joint.batch(batch_size=3, drop_remainder=False)
batch_x, batch_y = next(iter(ds))
print('Batch-x: \n', batch_x.numpy())
```

Batch-x:

```
[[0.165 0.901 0.631]
 [0.435 0.292 0.643]
 [0.976 0.435 0.66 ]]
```

```
print('Batch-y: ', batch_y.numpy())
```

Batch-y: [0 1 2]

TensorFlow-1: Basic

C.3. Shuffle, Batch dan Repeat

- Untuk mengulangi dataset yang telah ter-batch sebanyak dua kali

```
ds = ds_joint.batch(3).repeat(count=2)
for i,(batch_x, batch_y) in enumerate(ds):
    print(i, batch_x.shape, batch_y.numpy())
```

```
0 (3, 3) [0 1 2]
1 (1, 3) [3]
2 (3, 3) [0 1 2]
3 (1, 3) [3]
```

- Kode di atas menghasilkan dua copy dari setiap batch. Jika urutan diubah maka hasil berbeda

```
ds = ds_joint.repeat(count=2).batch(3)
for i,(batch_x, batch_y) in enumerate(ds):
    print(i, batch_x.shape, batch_y.numpy())
```

```
0 (3, 3) [0 1 2]
1 (3, 3) [3 0 1]
2 (2, 3) [2 3]
```

TensorFlow-1: Basic

C.3. Shuffle, Batch dan Repeat

- Dua cell program di atas menunjukan
 - Jika batch diikuti repeat maka hasil 4 batch
 - Ketika repeat diikuti batch maka hasil 3 batch
- Untuk pemahaman lebih lanjut dilakukan tiga operasi yang diurut berbeda
 - Urutan: shuffle, batch, repeat

```
tf.random.set_seed(1)
## Order 1: shuffle -> batch -> repeat
ds = ds_joint.shuffle(4).batch(2).repeat(3)

for i,(batch_x, batch_y) in enumerate(ds):
    print(i, batch_x.shape, batch_y.numpy())
```

```
0 (2, 3) [2 1]
1 (2, 3) [0 3]
2 (2, 3) [0 3]
3 (2, 3) [1 2]
4 (2, 3) [3 0]
5 (2, 3) [1 2]
```

TensorFlow-1: Basic

C.3. Shuffle, Batch dan Repeat

- Urutan: batch, shuffle, repeat

```
tf.random.set_seed(1)
## Order 2: batch -> shuffle -> repeat
ds = ds_joint.batch(2).shuffle(4).repeat(3)
for i,(batch_x, batch_y) in enumerate(ds):
    print(i, batch_x.shape, batch_y.numpy())
```

0	(2, 3)	[0 1]
1	(2, 3)	[2 3]
2	(2, 3)	[0 1]
3	(2, 3)	[2 3]
4	(2, 3)	[2 3]
5	(2, 3)	[0 1]

- Untuk urutan pertama shuffle, batch, repeat → dataset tershuffle seperti yang diharapkan
- Untuk urutan kedua batch, shuffle, repeat → elemen-elemen batch tidak tershuffle sama sekali
- Bagaimana hasil jika digunakan operasi shuffle setelah repeat? Mis: batch, repeat, shuffle

TensorFlow-1: Basic

C.4. Membuat Dataset dari file-file

- Kita akan membuat dataset berasal dari file-file image yang disimpan pada disk
 - Pasikan folder tempat file program anda terdapat folder cat_dog_images yang berisi 6 file image kucing dan anjing dalam format JPEG
 - Akan dibuat dengan menggunakan dua modul tambahan pada TensorFlow yaitu **tf.io** untuk membaca isi file image dan **tf.image** untuk mendekodekan isi mentah dan merubah ukuran (*resize*) image
- Untuk mempelajari fungsi-fungsi tambahan dari tf.io dan tf.image dapat diakses pada link berikut:

https://www.tensorflow.org/api_docs/python/tf/io

https://www.tensorflow.org/api_docs/python/tf/image

TensorFlow-1: Basic

C.4. Membuat Dataset dari file-file

- Terlebih dahulu melihat isi folder cat_dog_images, dengan librari pathlib

```
import pathlib
imgdir_path = pathlib.Path('cat_dog_images')
file_list = sorted([str(path) for path in imgdir_path.glob('*.*')])
print(file_list)
```

```
['cat_dog_images/cat-01.jpg', 'cat_dog_images/cat-02.jpg',
'cat_dog_images/cat-03.jpg', 'cat_dog_images/dog-01.jpg',
'cat_dog_images/dog-02.jpg', 'cat_dog_images/dog-03.jpg']
```

TensorFlow-1: Basic

C.4. Membuat Dataset dari file-file

- Untuk visualisasikan file-file image tersebut

```
import matplotlib.pyplot as plt
import os

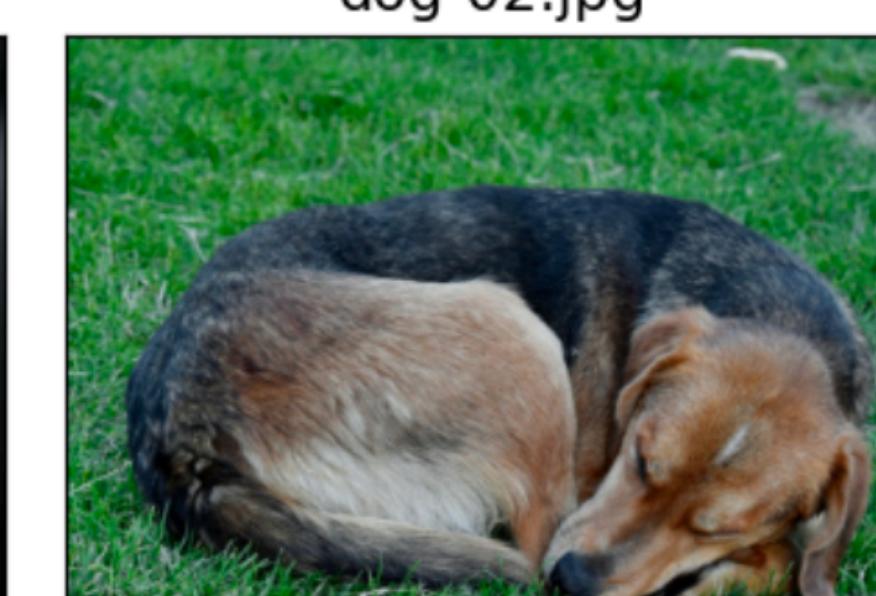
fig = plt.figure(figsize=(10, 5))

for i,file in enumerate(file_list):
    img_raw = tf.io.read_file(file)
    img = tf.image.decode_image(img_raw)
    print('Image shape: ', img.shape)
    ax = fig.add_subplot(2, 3, i+1)
    ax.set_xticks([]); ax.set_yticks([])
    ax.imshow(img)
    ax.set_title(os.path.basename(file), size=15)

# plt.savefig('ch13-catdot-examples.pdf')
plt.tight_layout()
plt.show()
```

```
Image shape: (900, 1200, 3)
Image shape: (900, 1200, 3)
Image shape: (900, 742, 3)
Image shape: (800, 1200, 3)
Image shape: (800, 1200, 3)
Image shape: (900, 1200, 3)
```

Terlihat image-image mempunyai aspect ratio berbeda



TensorFlow-1: Basic

C.4. Membuat Dataset dari file-file

- Akan dibuat aspect ratio menjadi sama dan label-label image disediakan di nama-nama file (label 1 = anjing, label 0 = kucing)

```
labels = [1 if 'dog' in os.path.basename(file) else 0
          for file in file_list]
print(labels)
```

[0, 0, 0, 1, 1, 1]

Sudah dipunyai dua list: list nama file dan list dari label-label

- Dibuat joint dataset untuk kedua list

```
ds_files_labels = tf.data.Dataset.from_tensor_slices((file_list, labels))

for item in ds_files_labels:
    print(item[0].numpy(), item[1].numpy())
```

b'cat_dog_images/cat-01.jpg' 0
b'cat_dog_images/cat-02.jpg' 0
b'cat_dog_images/cat-03.jpg' 0
b'cat_dog_images/dog-01.jpg' 1
b'cat_dog_images/dog-02.jpg' 1
b'cat_dog_images/dog-03.jpg' 1

TensorFlow-1: Basic

C.4. Membuat Dataset dari file-file

- Dataset akan ditransformasi: load isi image, dekodekan dan resize ke ukuran yang diinginkan (mis. 80×120)
- Kita perlu mengamplikasikan beberapa preprocessing sekaligus, maka dibuat *helper function* dan digunakan saat memanggil `.map()`
- Kode akan menghasilkan ukuran image yang sama

```
def load_and_preprocess(path, label):
    image = tf.io.read_file(path)
    image = tf.image.decode_jpeg(image, channels=3)
    image = tf.image.resize(image, [img_height, img_width])
    image /= 255.0

    return image, label

img_width, img_height = 120, 80

ds_images_labels = ds_files_labels.map(load_and_preprocess)

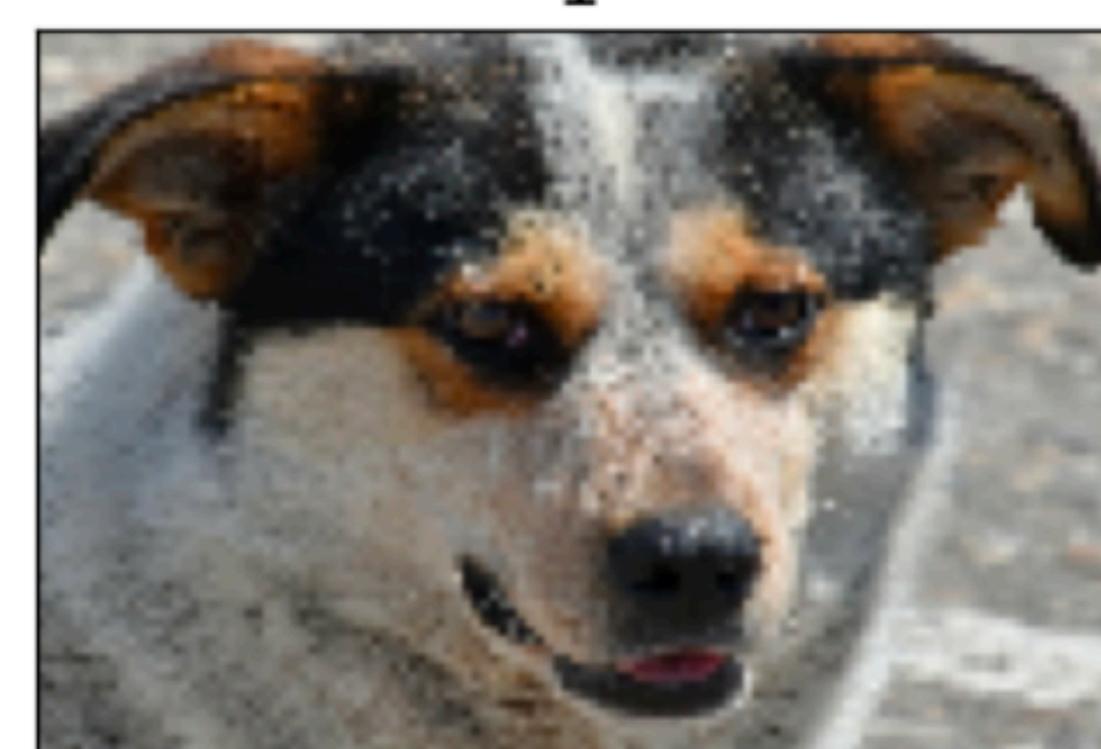
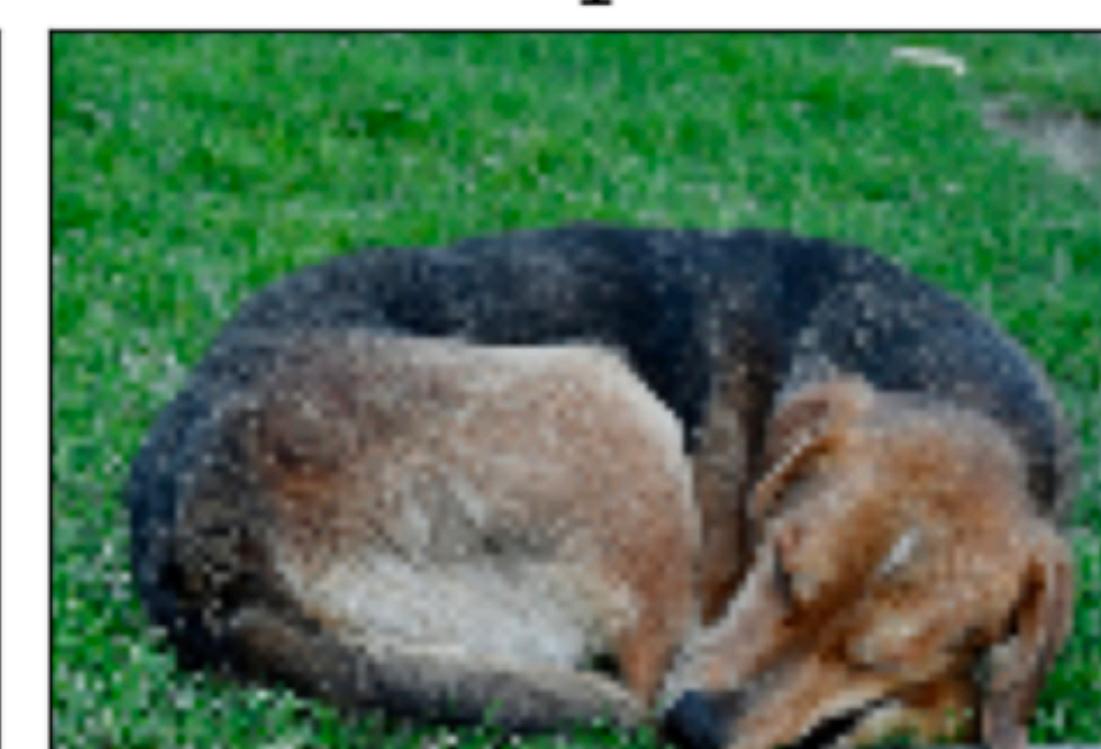
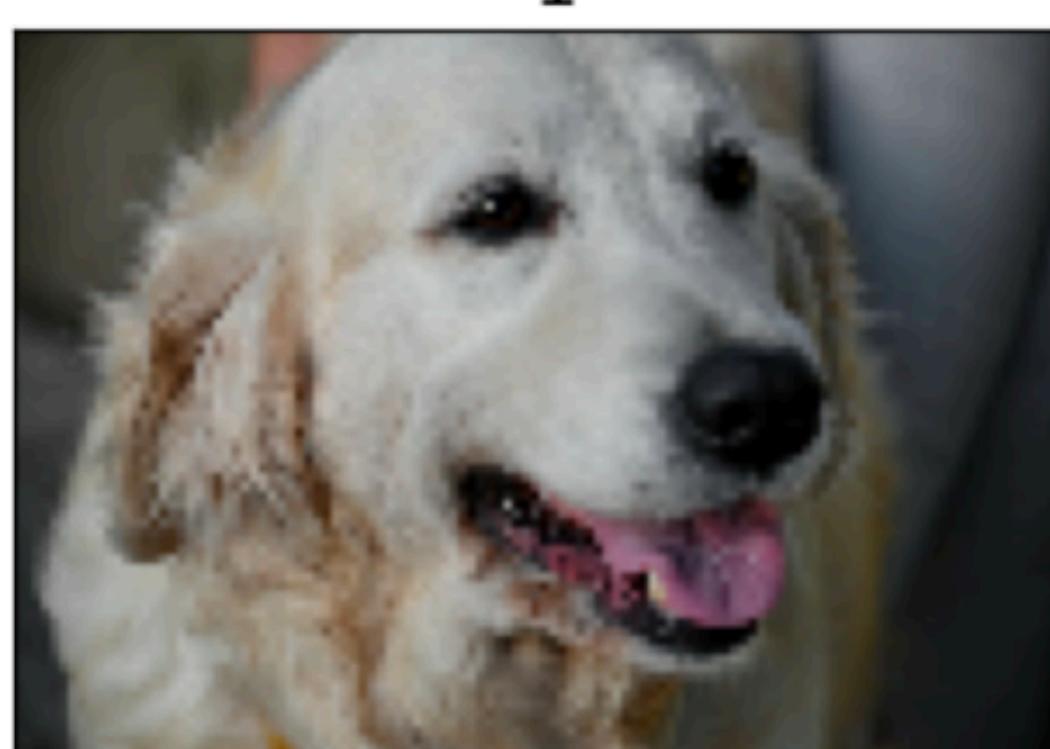
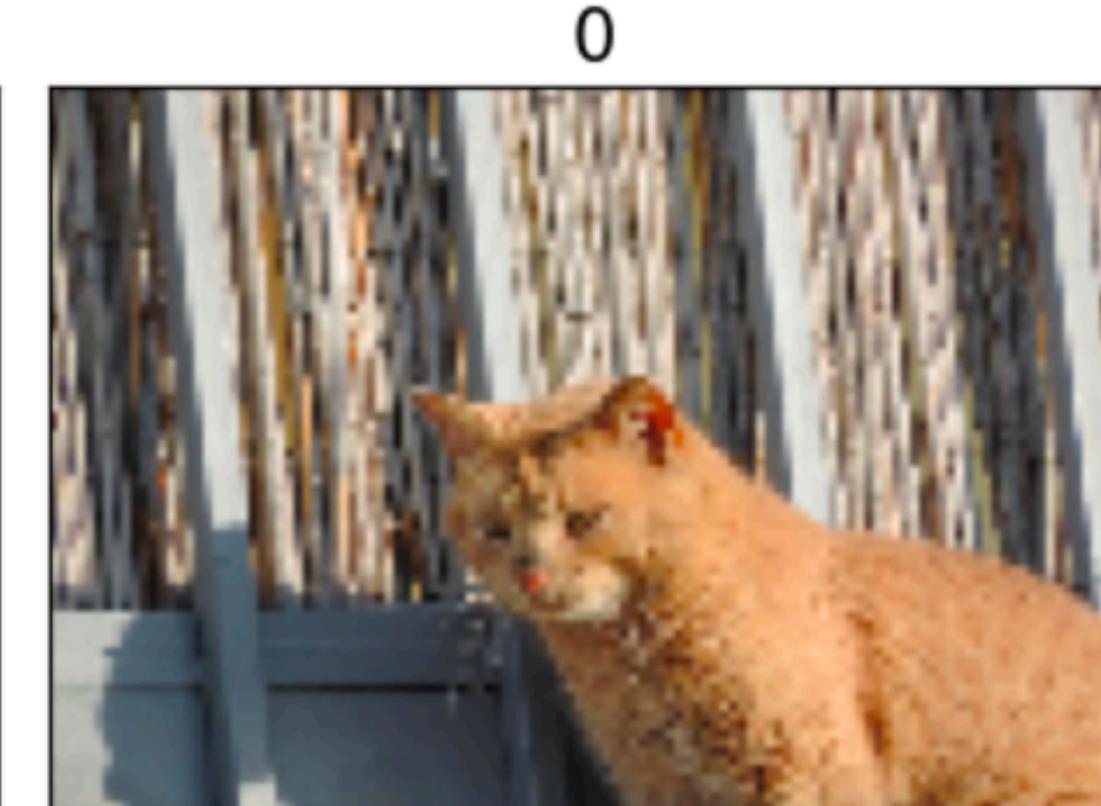
fig = plt.figure(figsize=(10, 5))
for i,example in enumerate(ds_images_labels):
    print(example[0].shape, example[1].numpy())
    ax = fig.add_subplot(2, 3, i+1)
    ax.set_xticks([]); ax.set_yticks([])
    ax.imshow(example[0])
    ax.set_title('{}'.format(example[1].numpy())),
                           size=15)

plt.tight_layout()
# plt.savefig('ch13-catdog-dataset.pdf')
plt.show()
```

TensorFlow-1: Basic

C.4. Membuat Dataset dari file-file

```
(80, 120, 3) 0
(80, 120, 3) 0
(80, 120, 3) 0
(80, 120, 3) 1
(80, 120, 3) 1
(80, 120, 3) 1
```



TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

- Librari tensorflow_datasets menyediakan banyak koleksi yang bagus dan tersedia secara gratis untuk melakukan training atau mengevaluasi model-model *deep learning*
- Dataset yang disediakan sudah diformat dengan baik dan mempunyai deskripsi yang informatif
- Dataset-dataset ini semuanya dipersiapkan untuk digunakan sebagai objek `tf.data.Datasets` sehingga fungsi-fungsi yang dijelaskan sebelumnya dapat digunakan
- Overview dataset yang disediakan TensorFlow dapat diakses pada link berikut

https://www.tensorflow.org/datasets/catalog/overview#all_datasets

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

- Pertama install librari tensorflow_datasets

```
! pip install tensorflow-datasets
```

- Import modul dan juga lihat list dataset yang disediakan

```
import tensorflow_datasets as tfds
print(len(tfds.list_builders()))
print(tfds.list_builders()[:5])
```

```
1139
['abstract_reasoning', 'accentdb', 'aeslc', 'aflw2k3d', 'ag_news_subset']
```

- Terdapat 1139 datasets (bisa jadi bertambah), dan menampilkan 5 nama dataset teratas yang ada

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

- Ada dua cara untuk fetching dataset, dan akan ditunjukan keduanya dengan fetching dua dataset berbeda, yaitu CelebA (`celeb_a`) dan MNIST

1. Cara Pertama:

Pendekatan ini mempunya tiga langkah

- Memanggil fungsi builder
- Mengsekusi metode `download_and_prepare()`
- Memanggil metode `as_dataset()`

- Dilihat full list dataset yang ada

```
## Run this to see the full list:
tfds.list_builders()
```

```
celeba_bldr = tfds.builder('celeb_a')
```

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

- Akan menggunakan dataset CelebA dan mencetak deskripsi terkait

```
print(celeba_bldr.info.features)
```

```
print(celeba_bldr.info.features.keys())
```

```
dict_keys(['image', 'landmarks', 'attributes'])
```

```
print(celeba_bldr.info.features['image'])
```

```
Image(shape=(218, 178, 3), dtype=uint8)
```

```
print(celeba_bldr.info.features['attributes'].keys())
```

```
dict_keys(['5_o_Clock_Shadow', 'Arched_Eyebrows', 'Attractive',
'Bags_Under_Eyes', 'Bald', 'Bangs', 'Big_Lips', 'Big_Nose', 'Black_Hair',
'Blond_Hair', 'Blurry', 'Brown_Hair', 'Bushy_Eyebrows', 'Chubby', 'Double_Chin',
'Eyeglasses', 'Goatee', 'Gray_Hair', 'Heavy_Makeup', 'High_Cheekbones', 'Male',
'Mouth_Slightly_Open', 'Mustache', 'Narrow_Eyes', 'No_Beard', 'Oval_Face',
'Pale_Skin', 'Pointy_Nose', 'Receding_Hairline', 'Rosy_Cheeks', 'Sideburns',
'Smiling', 'Straight_Hair', 'Wavy_Hair', 'Wearing_Earrings', 'Wearing_Hat',
'Wearing_Lipstick', 'Wearing_Necklace', 'Wearing_Necktie', 'Young'])
```

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

```
print(celeba_bldr.info.citation)

@inproceedings{conf/iccv/LiuLWT15,
    added-at = {2018-10-09T00:00:00.000+0200},
    author = {Liu, Ziwei and Luo, Ping and Wang, Xiaogang and Tang, Xiaoou},
    biburl =
{https://www.bibsonomy.org/bibtex/250e4959be61db325d2f02c1d8cd7bfbb/dblp},
    booktitle = {ICCV},
    crossref = {conf/iccv/2015},
    ee = {http://doi.ieeecomputersociety.org/10.1109/ICCV.2015.425},
    interhash = {3f735aaa11957e73914bbe2ca9d5e702},
    intrahash = {50e4959be61db325d2f02c1d8cd7bfbb},
    isbn = {978-1-4673-8391-2},
    keywords = {dblp},
    pages = {3730-3738},
    publisher = {IEEE Computer Society},
    timestamp = {2018-10-11T11:43:28.000+0200},
    title = {Deep Learning Face Attributes in the Wild.},
    url = {http://dblp.uni-trier.de/db/conf/iccv/iccv2015.html#LiuLWT15},
    year = 2015
}
```

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

- Perintah info-info di atas digunakan untuk mengetahui struktur dari dataset
- Feature yang tersimpan pada sebuah dictionary terdiri dari 3 key:
 - ➔ image: menyatakan image muka dari selebriti
 - ➔ landmarks: menyatakan dictionary dari titik ekstaksi wajah, seperti posisi mata, hidung, dsb
 - ➔ attributes: dictionary dari 40 attribute orang pada image, seperti ekspresi wajah, makeup, properti rambut, dsb
- Selanjutnya memanggil metode `download_and_prepare()`

```
# Download the data, prepare it, and write it to disk
celeba_bldr.download_and_prepare()
```

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

- Melakukan instantiate (membuat objek) dari dataset dengan nama CelebData\

```
# Load data from disk as tf.data.Datasets
CelebData = celeba_bldr.as_dataset(shuffle_files=False)
CelebData.keys()

dict_keys(['train', 'validation', 'test'])
```

Terlihat bahwa data sudah terbagi menjadi train, test dan validation

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

- Untuk melihat image, sebelumnya kode berikut dapat dieksekusi

```
ds_train = CelebData['train']
assert isinstance(ds_train, tf.data.Dataset)

example = next(iter(ds_train))
print(type(example))
print(example.keys())
```

```
<class 'dict'>
dict_keys(['attributes', 'image', 'landmarks'])
```

Terlihat elemen berbentuk dictionary, maka harus diformat ke dalam bentuk tuple (features, label)

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

- Untuk label kita gunakan kategori ‘Male’ dari attribute dengan transforms via .map()

```
ds_train = ds_train.map(lambda item:  
    (item['image'], tf.cast(item['attributes']['Male'], tf.int32)))
```

- Selanjutnya dataset di-batch dan mengambil 18 examples darinya

```
: ds_train = ds_train.batch(18)  
images, labels = next(iter(ds_train))  
  
print(images.shape, labels)
```

```
(18, 218, 178, 3) tf.Tensor([0 1 0 0 1 1 1 1 0 0 0 1 0 0 1 1 1], shape=(18,),  
dtype=int32)
```

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

```
fig = plt.figure(figsize=(12, 8))
for i,(image,label) in enumerate(zip(images, labels)):
    ax = fig.add_subplot(3, 6, i+1)
    ax.set_xticks([]); ax.set_yticks([])
    ax.imshow(image)
    ax.set_title('{}'.format(label), size=15)

plt.show()
```



Diperoleh 18 image selebriti dengan label 1 untuk laki-laki dan label 0 untuk perempuan

Proses fetching dan using dataset CelebA telah dilakukan

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

2. Cara Kedua:

- Melakukan pendekatan kedua untuk fetching dataset MNIST
- Terdapat fungsi wrapper yaitu `load()`, yang mengkombinasikan tiga langkah sebelumnya

```
mnist, mnist_info = tfds.load('mnist', with_info=True, shuffle_files=False)
```

```
print(mnist_info)
```

```
print(mnist.keys())
```

```
tfds.core.DatasetInfo(  
    name='mnist',  
    full_name='mnist/3.0.1',  
    description="""  
The MNIST database of handwritten digits.  
""",  
    homepage='http://yann.lecun.com/exdb/mnist/',  
    data_path='/Users/fikysuratman/tensorflow_datasets/mnist/3.0.1',  
    file_format=tfrecord,  
    download_size=11.06 MiB,
```

```
dataset_size=21.00 MiB,  
features=FeaturesDict({  
    'image': Image(shape=(28, 28, 1), dtype=uint8),  
    'label': ClassLabel(shape=(), dtype=int64, num_classes=10),  
}),  
supervised_keys=('image', 'label'),  
disable_shuffling=False,  
splits={  
    'test': <SplitInfo num_examples=10000, num_shards=1>,  
    'train': <SplitInfo num_examples=60000, num_shards=1>,  
},  
citation="""@article{lecun2010mnist,  
    title={MNIST handwritten digit database},  
    author={LeCun, Yann and Cortes, Corinna and Burges, CJ},  
    journal={ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist},  
    volume={2},  
    year={2010}  
}""",  
)  
dict_keys(['test', 'train'])
```

Dataset MNIST terbagi menjadi
dua partisi `train` dan `test`

TensorFlow-1: Basic

C.5. Fetching Dataset dari librari tensorflow_datasets

- Kita akan mengambil partisi train, dan mentransformasikan elemen berbentuk dictionary menjadi tuple, dan memvisualisasikan 10 contoh data

```
ds_train = mnist['train']
assert isinstance(ds_train, tf.data.Dataset)

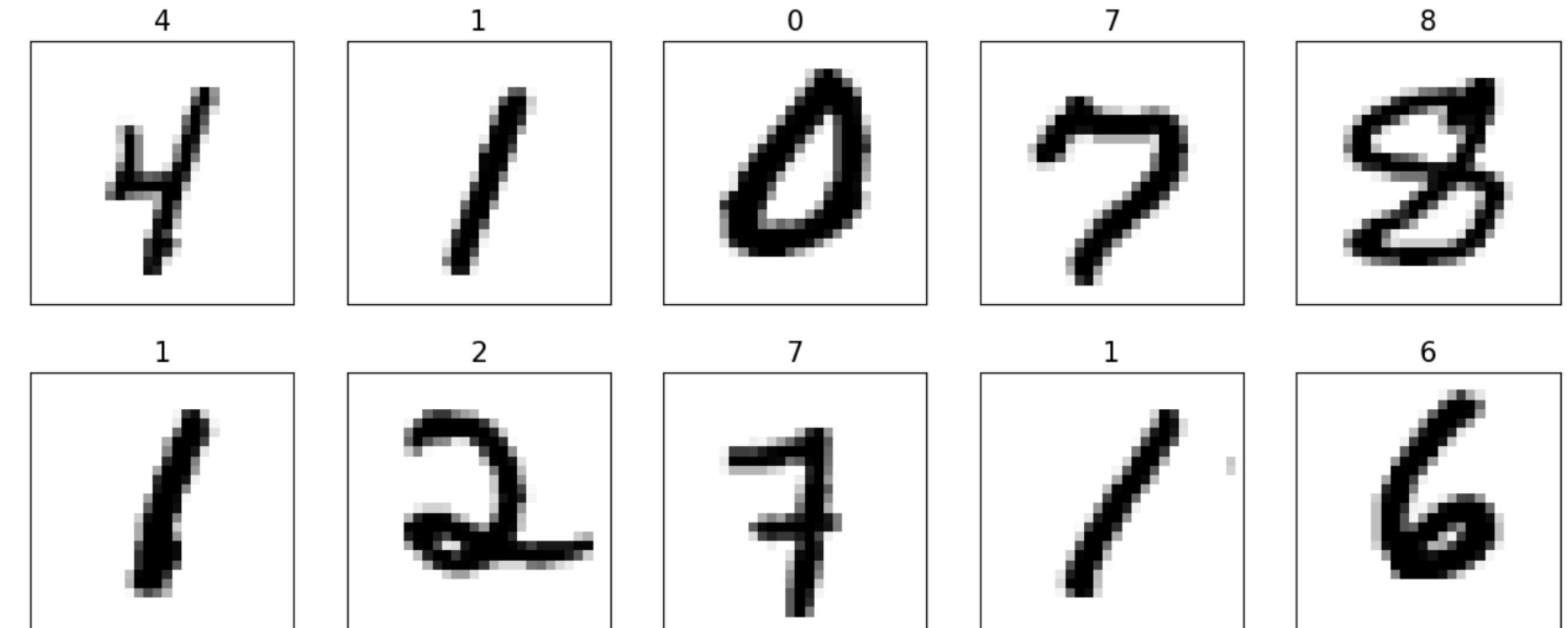
ds_train = ds_train.map(lambda item: (item['image'], item['label']))

ds_train = ds_train.batch(10)
batch = next(iter(ds_train))
print(batch[0].shape, batch[1])

fig = plt.figure(figsize=(15, 6))
for i,(image,label) in enumerate(zip(batch[0], batch[1])):
    ax = fig.add_subplot(2, 5, i+1)
    ax.set_xticks([]); ax.set_yticks([])
    ax.imshow(image[:, :, 0], cmap='gray_r')
    ax.set_title('{}\n{}'.format(label), size=15)

plt.show()
```

(10, 28, 28, 1) tf.Tensor([4 1 0 7 8 1 2 7 1 6], shape=(10,), dtype=int64)



**Memanipulasi dataset dan fetching dataset dari
librari tensorflow_datasets telah selesai**

**Pada bagian selanjutnya akan dilihat bagaimana
membangun model Neural Networks di TensorFlow**

Thank You